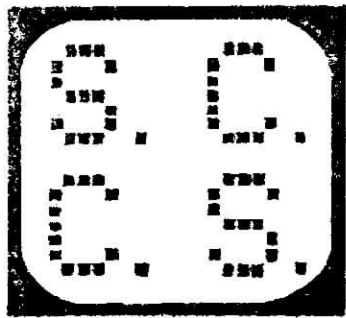




INTERFACE

....a newsletter for computer hobbyists
designed to connect people and ideas...

SEPTEMBER, 1975
VOL 1, #1

UPDATE

"BINGO" SURVEY

D. Ekstrand's "bingo-card" survey at the last meeting has given us a lot of valuable data. (We may well have set an excellent precedent for data and voting tally!) Following is a summary of the results:

Out of 184 cards turned in, 62 showed ownership of microcomputers and 29 of minicomputers. Interest was expressed in the following: Altair (73), DEC (10), Nova (31), Data Point (14), Microprocessors (50), Music/Synthesizers (37), CRTs (79), TTYs (53), printers (51), tape & card readers (49), TVs (69), modems (54), UARTs (50), memory (62), audio cassettes (56), cassette drives (56), floppy discs (69), disc drives (47); Basic (90), Fortran (54), Cobol (20), PL1 (33), APL (35), assembly language (61); games (84), business applications (63), scientific applications (68), operating systems (70), operating utilities (44); LSI-11 (20), ROMs (28), group purchases (80), and 13 people say they've "got a deal" (in the computer field, we assume!).

When it comes to setting standards, the following areas found support: documentation (49), hardware (23), software (30), interface (24), graphics (9).

The following would like help with: hardware (26), software (37), hardware purchase (40), software purchase (21). Interested in swapping are: hardware (39), software (57), operating systems (36).

Offering help to SCCS in the following areas are: as officer (11), as director (21), with

SOUTHERN CALIFORNIA
COMPUTER SOCIETY
(SCCS)
P.O. BOX 987
SOUTH PASADENA, CA.
91030
(213) 632-3108

membership (19), with the newsletter (20), with public relations (7), as speakers (7), with finding meeting places (5). (We plan to hold you all to it!)

On opinions: 64 preferred Sunday as the meeting day; 98 wanted Saturday and 10 were for weekday evenings. 110 preferred club meetings as we've been having, and 30 wanted smaller groups. On area preferences we found: Van Nuys (57), Glendale/Pasadena (23), LA/USC (41), Westwood/UCLA (51), South Bay (63). 26 wanted an all-out effort to increase membership, and 99 preferred growing slowly. 101 were in favor of continuing refreshments on a "contribution" basis, and 16 would like it more organized.

And finally, 35 of you are already members of some computer organization or another. A last note: only 5 put "Ms" on the card (as against "Mr") -- come on, ladies, let's balance out this organization!

What's being done about all this? Well, D. Ekstrand plans to coordinate a lot of "needers" with "havers"; George Tate is looking up those who want to be officers or directors; Ron Carlson will add this information to the rest for his data bank; Jeff Landry, Johnnie Keim and Roy Fultun will be looking things over for INTERFACE contributors; and surely others will scan the printouts for various and sundry reasons.

All this will, of course, take time. In the meantime, if you need the specifics on any particular item (we have names and phone numbers behind these figures) you can try calling me (Jon Walden: 769-6569). If I'm not out getting an instantaneous lobotomy, I'll try to help!

VOTING

To be sure everybody understands what's happening: At the last meeting a set of tentative by-laws was passed out to everybody, and the procedure for recommending changes was announced by Jerry. In keeping with this, the steering committee will submit the revised by-laws at the next meeting, and the matter will be put to a vote. All who are present at the meeting may vote on the by-laws.

If the by-laws are accepted, however, they will immediately take effect; election of officers and directors will follow. Only those present who are paid members will be allowed to vote on these matters.

Upon acceptance of the by-laws, we are a formal society with defined membership. To be assured of your vote, get your dues to Hal Lashlee before this meeting. You may use the form printed on the back of INTERFACE if you like, but be sure to do it now.

8008 RAP SESSIONS

All you 8008 people, take heart! Things are not passing you by! Derek McColl has a homebrew working system built around the 8008; he's got 8K memory and a working 16-level hardware push-pop stack. And he wants to set up rap sessions in his home.

Derek says his expertise is mainly in hardware and assembly

language. He wants to get Basic up for 8008 users and hopes people with knowledge of both Basic and interpreters will show up.

The sessions hopefully will cover all sorts of interests. If you're into this, give Derek a call at 379-9634 (after 6PM). He plans to meet in groups of 10 to 12 people, and he'll get back to you on date and time.

(Thanks, Derek. That's what this club and newsletter are all about!)

GROUP PURCHASE by Hal Lashlee

1) Processor Technology: 15% discount on a \$1500.00 order. Order and check made payable to "Southern California Computer Society -- Trust A" must be received on or before Sept. 15; pick-up at the 20th meeting. Our last order was for approximately \$2000.00 and everything has been delivered except mother boards and card guides. PS: 4K kits are \$215.00 effective 9-1-75. Please include sales tax and \$1 for handling.

2) LSI-11: Oct. 15, 975, will be the order date with delivery (hopefully) around the first of the year. If everyone that expressed an interest does order, then the order is on. Please do not send money yet. Because of the large amount of money involved Pearce Young and myself are obtaining a bond in the amount of \$100,000.00. Everyone expressing an interest will be contacted with complete details.

3) Decwriters (LA-36): We are hoping to include at the same time as the LSI-11 with approx. one month delivery. If you are interested please contact me before Oct. 1.

4) Altair CPU PC boards: \$15.00. Call, write or see me at the meeting.

5) What do you want or need? I would like suggestions, sources and any other ideas and help you might give.

Contact Hal Lashlee at the club address, or phone on any of the above.

NEW MEMBERS

PLEASE INCLUDE YOUR COMPLETE ADDRESS (WITH ZIP) AND PHONE NUMBER WITH YOUR \$10.00 DUES.

OLD MEMBERS

A COMPLETE LIST OF ALL MEMBERS WILL BE AVAILABLE AT THE NEXT MEETING. IF YOU DO NOT WANT YOUR PHONE NUMBER PUBLISHED, PLEASE CONTACT HAL LASHLEE BEFORE IT IS TOO LATE!

INTERFACIAL

Several of you expressed interest in designing a masthead for INTERFACE. We'll rotate through a few of these, look for your comments, and try to settle on an official one by the first of the year.

For this one, thanks goes to Mike Stern, who got it to us ready for reduction within three days after the plea at our last meeting!

Did you hear about the general who asked the computer, "Will it be peace or war?"

The computer replied, "Yes."

"Yes what?" the general yelled.

"Yes, SIR!" came the answer.

CLUB TREASURY REPORT

INCOME:

Dues: (112 members)	\$1120.00
Donations:	35.00
	<u>\$1155.00</u>

EXPENSES:

First meeting notice	\$ 16.94
First newsletter	73.12
Supplies	1.89
	<u>91.95</u>

BALANCE:

\$1063.05
(This does not include the cost of printing the "bingo cards" nor the postage of the second newsletter. Printing for the second newsletter has been done without cost to the club.)

DREAMS AND IDEAS

APPLICATION

If you're looking for something to do with your system, think about this: Nelson, in Computer Lib (see BOOK REVIEW, p. 2), notes that a 33 Teletype with an adapter kit for punching Braille can make copies of anything you have in memory! Anybody interested in trying to set this up?

BYTE

It's out! Issue #1 (Sept. 1975) of Byte. \$1.50/issue; still \$10/yr. to become a charter subscriber (BankAmericard or Master Charge OK). Will soon be \$12/yr.

It's like old-home-week: MITS, Godbout, Processor Technology, the Mike family, Scelbi, Sphere, PCC, TCH, Homebrew, Micro-8, Digital Group...they're all there.

And a fine collection of articles (considering 7 weeks from inception to press!): summary of microprocessors, the RGS 008A kit, how to recycle used ICs and analyze surplus keyboards and write your own assembler, a very informative thing on serial interface, a game called LIFE.... and more....

It's inconceivable that any computer hobbyist would want to be without this magazine. (I know how many newsletters you're already getting; I'm talking about 96 pages of dynamite!)

Once again:

BYTE

Green Publishing, Inc.
Peterborough, N.H.

03458

(603) 924-3873

To quote from page 5: "Sophisticated fun requires sophisticated thought and hard work." The staff of BYTE have set a standard.....

COMPUTER BITS

Jerry Ogden has begun a column in Popular Electronics called "Computer Bits". In the September issue (page 57) he suggests a badly needed "Hobbyist Interchange Tape System" (HIT). The article, complete with programs, deserves study -- and Jerry, for his fine column, deserves our thanks.



.....BOOK REVIEW.....
by Jon Walden

If you've watched Hal Lashlee's activities lately, you'll understand why he asked me if I could get in the review he promised! He's been a one-man ant hill lately; we'll let him off the

hook this time!

Actually I'm delighted to have the chance to sing out about Theodor H. Nelson's book, "Computer Lib", and its flip side, "Dream Machines". I could wax eloquent with superlatives (and I probably will)! Not only about the inviting cameos ("articles" or "chapters" would fail to reflect the grab this book has) which range from the elementary to the profound and cover more aspects of the field than I had ever suspected to exist but also about the spirit of this book.

It's the man's attitude, his vision, his head that spaces me. He leaves me high, in the finest sense of that word. Let me throw some quick quotes at you:

There is no question of whether the computer will remake society; it has.

"Rigid and inhuman" computer systems are the creation of rigid and inhuman people.

Microprocessors are what's happening.

The technicalities matter a lot, but the unifying vision matters more.

Add to this a lot of solid information about microprocessors thru the bigies, languages, operating systems, time-sharing, peripherals, applications (many ideas to explore), and you can begin to see why I get so turned on...

What I would really like to do is just reprint the whole book for you! No? Then give up a few martinis or a pair of new jeans and shell out the seven bucks. Dick or Lois will sell it to you at the Computer Store or you can order it thru PCC. Whether you're just starting out or have been toying with computers since vacuum tube days, you'll never again wonder what to pick up when you find a few minutes to spare.

Only one caution: Keep a magnifying glass nearby; the data is packed in tight little characters that tend to run together far sooner than you're ready to quit reading!

Computer Lib? Ted Nelson defines it as "making people freer through computers. That's all."



INTERFACE NOTES

For the many comments and suggestions on our "announcement" issue...thanks. We hope to have an official Editor and staff soon. In the makeshift meantime the persons listed below have volunteered their time and energy to help get this show on the road:

Editorial committee:

D. Ekstrand
Roy Fultun
Roger Hall
Bruce Holmen
Johnnie Keim
Hal Lashlee
Jon Walden
Pearce Young

Hardware consultants:

Ruben Loshak
Owen Phairis
Carl David Todd

Software consultants:

Ron Carlson
Jeff Landry
Larry Press

Until formal policy is adopted, the following will stand in its stead:

1) After this issue, only SCCS members whose dues are paid will be assured of receiving INTERFACE.

2) We plan to shoot for a mailing on the first of each month. Anything coming in later than the 15th will be saved till next time. (The deadline is for EVERYBODY, folks -- no "in-group" exceptions from now on!)

3) No more quickie handwritten stuff. Type it. (See how fast we learn!)

4) We have gotten permission from several NIs to reprint their material. In keeping with this spirit of sharing, INTERFACE invites you to copy anything you find valuable in its pages. Help spread the goodies...

5) If you've got information but you aren't good at organizing or writing, we'll try to find you someone who is. Together we'll get it together.

Mailing address will temporarily be:

INTERFACE
11557 SUNSHINE TERRACE
STUDIO CITY, CA. 91604

For information, you can call Jon Walden: (213) 769-6569.

NOTE: You will help a lot by getting your articles and ads to me before the 15th deadline. The amount of work needed to get out even a small newsletter is more than you might imagine!

Oh yes....our first apology goes to Jerry Silver for spelling his name with a "G"! Sorry, Jerry!

.....LETTERS.....

Congratulations on your first issue of INTERFACE, also, for the three hole punch which will save time for....people with their filing.

Bill Pfeiffer

Congratulations and thanks to the whole staff for putting together a fine first issue of INTERFACE. In view of the short time you had for its preparation, you did a really phenomenal job!

Rudolf Hirschmann

...I've been thinking of a few of the sorts of things the newsletter might do...

1. Print articles on and ideas for applications. At the meeting in El Segundo you asked for volunteer consultants in hardware and software, however, I wonder if "software" isn't too narrow a term. "Applications" might be a broader term which would encourage articles on how we can use our computers as teaching devices, in our homes, as toys, in order to make money, etc. I would encourage such contributions as well as descriptions of specific programs and programming techniques.

2. The newsletter can be thought of as a "bulletin board" to assist in the matching of members needs and resources. Needs and resources should not be restricted to hardware, programs and data, we should include people resources as well. For instance, the newsletter might be useful for someone looking for a collaborator on a project. (see Larry's ad)

3. In addition to printing generally informative articles, the

INTERFACE

PAGE 4

Newsletter could serve as a more "interactive" information source on specific questions. For example, we could publish specific questions along with a phone number to call with the answers, or we might try some group design or problem solving through the newsletter.

I guess that in general, I see the newsletter as a possible medium for communication between individual members as well as a means for broadcasting information to everyone. It is probably best also to think of it as a local newsletter first and as a journal second.

Larry Press

(Well said, Larry. We hope the DREAMS & IDEAS section will be used to explore those many areas beyond hardware and software.)

The matter of bylaws, policies, and elections is at its very best dull business, but it is necessary business if we are to survive as a Society. At our next meeting all of this should be behind us and we may look forward to new leadership and a healthy existence.

Let me share with you some of my dreams. I foresee the day when we may become the leading amateur organization -- not only in size, but in the contribution that we will make to the computer arts and sciences. The surveys from our first meeting clearly indicate the diversity of talent and interest which exists among our members.

I foresee a computer center operated in conjunction with county or city government, providing time share and a permanent meeting place for our members. Above all I see the center as a place for the stimulation of young minds.

I look forward to technical sessions and classes organized for the benefit of all our members and appealing to their various and diverse interests. We should never become dedicated solely to hardware or software, but to the computer arts and sciences. Although we should seek to enrich our own lives and understanding through the Society, we should not forget our responsibility to the community in which we live.

The first issue of INTERFACE contains the promise of a major

publication which can be self-supporting and second to none among amateur journals. This alone will become one of our important contributions.

All of us are participating in a new and exciting adventure, one which is probably akin to that experienced in the early days of amateur radio. Each of us has something to offer if it is only the desire to listen and to learn. The energy and enthusiasm which has been generated over the past few months guarantee our future. Frankly, I am proud to be a part of that future.

Pearce Young

BEGINNERBITS with Jan

I know some of you, like me, are still trying to catch your breath. We've been bombarded with new vocabulary, new methods, new ideas and new facts till we're numb. Maybe you started out with programmable calculators like I did (where did all the labeled buttons go?); maybe you've been into a specialized computer field and now you need to broaden your scope; or maybe you just dived with abandon into this beautiful awesome world and your blinking little monster sits before you with all its grand potential while you haven't the foggiest idea how to make it dance!

Well, bunky, you're not alone! It's time to find out what you've always wanted to know about computers but were afraid to ask! And it's time to get some answers.

Beginnerbits will specialize in rudiments; no question is too simple-minded for us to tackle! Ask and ye shall be pleasantly surprised to find out how all the little pieces suddenly begin to fit into place. (IBM, you ain't runnin the store no more!)

"Kilo-" means thousand; "K" stands for kilo; but 1K doesn't equal 1,000. Why? Because computers are set up to operate around powers of 2, and 1,000 isn't a power of 2. 1,024 is, however, (2¹⁰) and 1K means the power of 2 closest to 1,000. So:

1K = 1,024	16K = 16,384
2K = 2,048	32K = 32,768
4K = 4,096	64K = 65,536
8K = 8,192	etc.

In our decimal number system it all comes out with not much pattern. But in binary (a number system based on 2) the pattern shows itself:

1K =	10,000,000,000
2K =	100,000,000,000
4K =	1,000,000,000,000
8K =	10,000,000,000,000
16K =	100,000,000,000,000
32K =	1,000,000,000,000,000
64K =	10,000,000,000,000,000

In octal (based on 2³) and in hexadecimal (based on 2⁴) the patterns are also there:

	OCT	HEX
1K =	2,000 =	400
2K =	4,000 =	800
4K =	10,000 =	1,000
8K =	20,000 =	2,000
16K =	40,000 =	4,000
32K =	100,000 =	8,000
64K =	200,000 =	10,000

The confusion comes from wanting to express "power of two" segments in decimal language. In a short time, the translation of numbers between one system and another will begin to make sense. Don't push it; let it sink in gradually!

Now, when you hear "8K", the next logical question is: 8K what? Bits or words? Time out for brief definitions:

Bit: One piece of binary information. On paper that means a 1 or a zero. In the computer it means the smallest (fundamental) element of memory. On most of your front panels it means one light, either on or off.

Word (or Byte): A fixed number (6, 8, 12, 16, etc.) of bits which is considered as a unit by the computer during processing. (Technically, there is a difference between "word" and "byte", but this will do for now.)

What does 8K bits mean then? It depends on the word length of your computer. For those with 8-bit words, 8K bits = 1K words. And 8K words = 64K bits. When you're looking at memory specs, be sure whether they're talking about bits or words!

While we're on bits, to store a letter of the alphabet or a special character (#, ?, etc.) a computer usually needs 8 bits. For example, the ASCII code for "Z" is 11011010. To store a decimal numerical digit, the computer can get by with using

only 4 bits. For example 83 can be stored in one 8-bit word as 10000011. The notation is called BCD (binary-coded decimal) and means that since 10002 = 810 and 00112 = 310 the word will be treated as 8310 instead of 13110 as it would otherwise be.

Since Beginnerbits assumes less than you might think (!), the "2" subscript means that 1000 and 0011 are binary numbers; the "10" means 8 and 3 are decimal numbers. "8" would mean octal and "16" hexadecimal, all of which we will explore later for those to whom it's still Greek!

A distinction that might clear away some more fog: When you're using Basic (or any other high-level language) you need it translated into your particular machine's language. An Interpreter is a program which translates each Basic instruction as it comes to it and carries it out one step at a time. A Compiler (which usually compiles in several stages) is a program

which translates your whole Basic program (the "source" program) into machine language (the "object" program) and stores it away for use later on. Most of us will be using an Interpreter until we have a disc or some other large memory device on our system.

And I guess that does it for the first try at Beginnerbits. Let's have your comments and questions for feedback. And all you smart dudes who think this sounds like kindergarten might tread gently.. us grass-roots explorers are what this home-computer revolution is really all about...

SOFTWARE

CODING FOR PUNCHING TAPE READABLE CHARACTERS

The following coding was submitted by Bill Roch. Use it to punch readable characters on your paper tape for leader identification or for fun.

A = @?II@	SPACE = @@@@@@
B = @?%Z@	! = @W@
C = @?IIR@	" = @C@C@
D = @?!!@	# = @J?J?J@
E = @?%I@	\$ = @R%?)P@
F = @?EEA@	% = @SK42@
G = @?!)H@	& = @^-4(@
H = @?DD?@	' = @C@
I = @?I@	(= @?I@
J = @P ?@	) = @I@
K = @?LR!@	* = @RL?LR@
L = @? @	= = @LLL@
M = @?BDB?@	- = @DDD@
N = @?BDH?@	@ = @\ "-.P@
O = @?!!?@	[= @?I@
P = @?IIF@	\ = @ADH @
Q = @?!!?@	+ = @\$. \$@
R = @?IY@	; = @?>@
S = @V+)?@	^ = @\$B?B\$@
T = @AA?AA@	] = @I?@
U = @? ?@	< = @LR!@
V = @OP PO@	> = @IR!@
W = @?PH?@	, = @X8@
X = @IRLR!@	. = @XX@
Y = @AB<BA@	? = @BA)F@
Z = @1)%@	/ = @HDA@
1 = @I? @	6 = @~%X@
2 = @9%?@	7 = @AA9C@
3 = @I%Z@	8 = @Z%Z@
4 = @HLJ?H@	9 = @F))~@
5 = @W%-Y@	0 = @~!!~@

BASIC LOADER

For Altair 8800 users, here's a program to help you get your Basic up and running:

PROGRAM NAME: BASBOT				PROGRAMMER: D.E. TARBELL				DATE: 7-20-75				PAGE 1 OF 1			
DESCRIPTION: LOADS BASIC USING ABSOLUTE FORMAT															
ADDRESS	MACHINE CODE				ASSEMBLY LANGUAGE			COMMENTS							
					LABEL	OP CODE	OPERAND								
017000	041	000	000	NOTYET	LXIH	0	LOAD H,L WITH STARTING ADDRESS (0)								
017003	333	000			IN	0	READ STATUS								
017005	346	001*			ANI	1	CLEAR ALL BUT BIT 0								
017007	302#	003	036		JNZ	NOTYET	WAIT IF NOT READY								
017012	333	001		WATCHR	IN	1	READ A BYTE OF LEADER								
017014	376	152			CPI	152	COMPARE TO 152 (SYNC BYTE)								
017016	302	003	036		JNZ	NOTYET	TRY AGAIN IF NOT SYNC BYTE								
017021	333	000			IN	0	READ STATUS								
017023	346	001*			ANI	1	CLEAR ALL BUT BIT 0								
017025	302#	021	036		JNZ	WATCHR	WAIT IF NOT READY								
017030	333	001			IN	1	READ A BYTE OF DATA								
017032	167				MOV	M,A	MOVE BYTE FROM A TO MEMORY								
017033	043			INX	H	INCREMENT MEMORY POINTER									
017034	303	021	036	JMP	WATCHR	READ ANOTHER BYTE									
PATCHES	OLD	NEW													
002152	201	170	}	ORIGINAL PATCHES				* THIS BYTE IS A MASK FOR THE STATUS BIT, AND MAY BE DIFFERENT FOR DIFFERENT INTERFACE CARDS.							
002230	015	017													
002244	015	017													
002204	002	200*	}	OUTPUT STATUS CHECK				# THIS IS THE CONDITIONAL JUMP FOR STATUS, AND MAY BE DIFFERENT (JZ OR JNZ) FOR DIFFERENT INTERFACE CARDS.							
002205	312	302#													
002217	040	001*	}	INPUT STATUS CHECK											
002220	312	302#													
002550	315	000	}	DELETED EXTRA READ											
002551	214	000													
002552	004	000													

* THIS BYTE IS A MASK FOR THE STATUS BIT, AND MAY BE DIFFERENT FOR DIFFERENT INTERFACE CARDS.

THIS IS THE CONDITIONAL JUMP FOR STATUS, AND MAY BE DIFFERENT (JZ OR JNZ) FOR DIFFERENT INTERFACE CARDS.

TABLES

Do you want INTERFACE to include tables? (e.g. ASCII Codes; Powers of Two; Binary, Octal and Hex addition and multiplication tables; Binary-Octal-Hex-Decimal conversion tables, etc.) If you have had trouble getting these or other tables, let us know. We'll INTERFACE your needs!

FLEX

by Roy Fultun

Flex: An alternative hexadecimal notation oriented towards people who like to think in base ten...

In order to do pencil and paper arithmetic calculations in hex, a certain amount of confusion is introduced in the process of converting to decimal, calculating, and reconvertng any remainder back to hex (mod sixteen). The confusion arises from the fact that in hex, the first letter 'A' of the alphabet is not assigned to eleven (ten plus one), but to ten itself.

When a person is used to thinking of 'A' as the first letter of the alphabet and 'E' as the fifth letter, hex becomes somewhat cumbersome when related to decimal notation. The reader might try to perform the hex calculation (D + C) in his head. 'D', the fourth letter of the alphabet, stands for 13 while 'C', the third letter, stands for 12. Well $13 + 12 = 25$ and $25 - 16 = 9$ with a carry of 1, so D + C = 19 in hex.

DEC	HEX	FLEX	DEC	HEX	FLEX
0	0	0	8	8	8
1	1	1	9	9	9
2	2	2	10	A	Z
3	3	3	11	B	A
4	4	4	12	C	B
5	5	5	13	D	C
6	6	6	14	E	D
7	7	7	15	F	E

With flex, the situation is simplified by using 'Z' to stand for ten (think 'zero' or 'zehn' if you've studied German). 'Z' is the last letter of the alphabet and is followed by 'A' if the alphabet is thought of as being a circular list. The correspondence of the five letters 'A' through 'E' and the integers eleven through fifteen is a natural one.

To perform (C + D) in flex, 'C', the third letter stands for thirteen while 'D' stands for fourteen (it's the fourth letter). Thus C + D flex is $13 + 14 = 27$ decimal and $27 - 16 = 11$ or A in flex so that the total calculation in flex is C + D = 1A. The only difficulty in the calculation is the mental subtraction of sixteen from twenty-seven. The conversion of the remainder to 'A' is trivial and natural.

In addition to ease of use, flex has one other decided advantage over hex; flex is easier to learn.



8008 TO 8080 CONVERSION CHART

This conversion chart uses a quasi numerical order for the 8008 instructions so that the chart may be used by 8080 users who want to translate available 8008 programs for their use. No attempt has been made to translate the "extra" 8080 program segments. Thanks to Joe Ringland whose chart on page II-4 of The Digital Group Clearinghouse (VOL 1, #1) made this one much easier to do!

8008		#	8080	
			BYTES	
LrI	0D6	2	0D6	MVlr
INr	*0D0	1	0D4	INRr
DGr	*0D1	1	0D5	DGRr
RST	0A5	1	3A7	RST
RLC	002	1	007	RLC
RRC	012	1	017	RRC
RAL	022	1	027	RAL
RAR	032	1	037	RAR
RET	0X7	1	311	RET
RFC	003	1	320	RNC
RFZ	013	1	300	RNZ
RFS (RFN)	023	1	360	RP
RFP (RFEP)	033	1	340	RPO
RTC	043	1	330	RC
RTZ	053	1	310	RZ
RTS (RTN)	063	1	370	RM
RTP (RTEP)	073	1	350	RPE
ADI	004	2	306	ADI
ACI	014	2	316	ACI
SUI	024	2	326	SUI
SBI	034	2	336	SBI
NDI	044	2	346	ANI
XRI	054	2	356	XRI
ORI	064	2	366	ORI
CPI	074	2	376	CPI

JMP	1X4	3	303	JMP
JFC	100	3	322	JNC
JFZ	110	3	302	JNZ
JFS (JFN)	120	3	362	JP
JFP (JFEP)	130	3	342	JPO
JTC	140	3	332	JC
JTZ	150	3	312	JZ
JTS (JTN)	160	3	372	JM
JTP (JTEP)	170	3	352	JPE
CAL	1X6	3	315	CALL
CFC	102	3	324	CNC
CFZ	112	3	304	CNZ
CFS (CFN)	122	3	364	CP
CFP (CFEP)	132	3	344	CPO
CTC	142	3	334	CC
CTZ	152	3	314	CZ
CTS (CTN)	162	3	374	CM
CTP (CTEP)	172	3	354	CPE

ADr	20S	1	20S	ADDr
ACr	21S	1	21S	ADCr
SUr	22S	1	22S	SUBr
SBr	23S	1	23S	SBBr
NDR	24S	1	24S	ANAr
XRr	25S	1	25S	XRAr
ORr	26S	1	26S	ORAr
CPr	27S	1	27S	CMPr

Lr1r2	3DS	1	0DS	MOVr1r2
HLT	*377	1	166	HLT
	000			
	001			

8008
INP 01 OOM MM1 (MMM--Input port)
OUT 01 RRM MM1 (RRMMM --- Output port with RR ≠ 0)

8080
IN 333 (B₂ = Input device #)
OUT 323 (B₂ = Output device #)

	S/D	8008	8080
	0	A	B
S = Source	1	B	C
D = Destination	2	C	D
	3	D	E
X = Don't care	4	E	H
	5	H	L
A = 0 through 7	6	L	M
	7	M	A

ALTAIR PRACTICE PROGRAMS

Here are a couple of practice programs for those of you who have your 8800 running with 4K memory and no interfacing yet. They're good studies if you're not too familiar with the instruction set, and goodness knows we're all tired of "Fool On The Hill"!

The first one below lets you look at the registers and the flag (status) word. Begin the program at address 000010 and examine the stack in addresses

000000 through 000007:

10	{061	LXI SP	{LOAD STACK
11	{010		{POINTER WITH
12	{000		{TOP OF STACK
			{PLUS ONE.
13	305	PUSH B	{PUSH REGIS-
14	325	PUSH D	{TERS AND THE
15	345	PUSH H	{FLAG WORD
16	365	PUSH SW	{ONTO STACK.
17	{303	JMP	{LOOP HERE
20	{017		{UNTIL PRO-
21	{000		{GRAM IS END-
			{ED BY STOP.

Examine 000010 and Run. Stop the program and examine the stack:

ADDRESS	CONTENTS
000000	Register B
000001	" C
000002	" D
000003	" E
000004	" H
000005	" L
000006	Accumulator
000007	Flag word

Remember in the flag word bit 1 will always be 1, and bits 3 and 5 will always be zero. Bit 0 will reflect the carry bit and bit 4 the auxiliary carry; bit 2 is the parity bit, 6 the zero bit and 7 the sign bit. You can use higher memory to set up the registers, go back and push them and check the stack to see if you did it right.

"Pop" can be used similarly if the stack pointer is loaded with the bottom address of the stack. The importance of the stack will become obvious if you try to "CALL" a subroutine without first loading the SP with an address in your memory! The "RETURN" address has to be put where you can find it later on; the stack is where it automatically goes. If the SP happens to contain an address beyond the limit of your memory you will be "returned" to address 177777 since non-existent memory is read as all ones!

The program below will load your memory above address 000020 in various ways. Begin depositing at address zero:

0	{001	LXI B	{LOAD REGISTER
1	{377		{PAIR B/C WITH
2	{000		{LAST ADDRESS
			{IN MEMORY.
3	{076	MVI A	{MOVE ALL ONES
4	{377		{INTO A.
5	002	STAX B	STORE ONES.
6	013	DCX B	DEC. ADDRESS.

7	171	MOV C → A	{SEE IF YOU'RE
10	{326	SUI A	{DOWN TO AD-
11	{017		{DRESS 17 YET.
12	{302	JNZ	{IF NOT, LOOP
13	{003		{BACK & GO ON.
14	{000		{(AT 000003)
15	{303	JMP	{IF SO, LOOP
16	{015		{HERE UNTIL
17	{000		{PROGRAM IS
			{STOPPED.

If you want to "clear" memory, change 3 from 076 to 000 (NOP) and 4 from 377 to 257 (XRA A). An "exclusive or" with itself will "clear" the accumulator.

To store the address number in the address itself, change 3 from 076 to 000 and 4 from 377 to 171 (MOV C to A). (If 2 is anything other than zero, i.e. if your memory is larger than 1K this won't work since a given address can hold only one word and addresses above 000377 require two.)

For exploring other instructions which will accomplish the same thing, try the following substitutions: 1) At 10 substitute 374 (CPI) for 326 or 2) in place of 7(171), 10(326) and 11(017) try:

7	{076	MVI A
10	{017	
11	271	CMP C

Unlike SUI, CMP doesn't change either register, only the status bits. In the example above it doesn't matter, but in many cases it will.

If you'd rather make the changes going "up" in memory rather than "down", change 1 from 377 to 020, change 6 from 013 to 003 (INX B) and 11 from 017 to 377. It will do the same thing except that address 377 will be missed.

Sometimes the simple explorations can lead to heavy insights. FOOL AROUND!

ADDRESS	←	OPERAND
177777	↔	377 377

The problem considered in this article is what happens when a 16-bit binary number is broken into two 8-bit sections and these sections are then converted to octal. The 8080 instruction set and the Altair computer are used in the examples, but the principles apply to any processor with an 8-bit memory word and a 16-bit address when octal encoding is used. The tables

which follow assume a limit of 64K words.

When using an 8080 jump or call instruction, the address is placed in bytes 2 and 3 of the instruction (low order in byte 2 and high order in byte 3). Thus 303, 003, 000 means an unconditional jump to address 000003. You may rightly wonder why the 2nd and 3rd bytes are reversed (this reversal isn't universal) but the relationship between them and the actual address is nevertheless apparent for the first 1/4K of memory: 303, 377, 000 means jump to actual address 000377. But what happens if you want to jump to address 000400? The obvious relationship is gone when we find we must use 303, 000, 001 to get there. Why?

The reason is that the address is expressed in a 6-digit octal number (representing 16 binary digits) while the operand for the jump instruction is being expressed in two 3-digit octal numbers (8 binary digits each). If you aren't up on this peculiarity, take a look at the address lights on the Altair front panel. They're arranged like this:

....

(5 groups of 3 with a single light at the left). With the groupings it's easy to read the address in octal; all lights on reads 177777, the last word in 64K.

But notice the line which underscores the right (low order) 8 bits. It's put there mainly to point out the eight data lights above, but it can help us to understand our problem. It leaves these 8 bits nicely grouped:

....

and shows clearly that 377 is as much as we can get into byte 2. The left (high order) 8 bits are left looking like this:

....

What we have to do is rearrange the groupings in our mind in order to get the right octal code for byte 3:

..+..+..

Now, address 000400 looks like this:

0 000 000 100 000 000

To get our numbers for bytes 2 and 3 we think:

0 000 000 100 000 000

which gives us 000 for byte 2 and 001 for byte 3. In Don's program on page 5 he jumps to address 003, 036. To find the actual address he's jumping to we can write out:

00 011 110 00 000 011

change it to:

0 001 111 000 000 011

and read it as 017003, which it is as his "NOTYET" label shows.

Or, for a short-cut, we can use tables like these following:
(A = any number)

B2	B3	ADDRESS	B2	B3	ADD.
0AA, 000	00AA	00AA	0AA, 010	40AA	
1AA, 000	01AA	1AA, 010	1AA, 010	41AA	
2AA, 000	02AA	2AA, 010	2AA, 010	42AA	
3AA, 000	03AA	3AA, 010	3AA, 010	43AA	
0AA, 001	04AA	0AA, 011	0AA, 011	44AA	
1AA, 001	05AA	1AA, 011	1AA, 011	45AA	
2AA, 001	06AA	2AA, 011	2AA, 011	46AA	
3AA, 001	07AA	3AA, 011	3AA, 011	47AA	
0AA, 002	10AA	0AA, 012	0AA, 012	50AA	
1AA, 002	11AA	1AA, 012	1AA, 012	51AA	
2AA, 002	12AA	2AA, 012	2AA, 012	52AA	
3AA, 002	13AA	3AA, 012	3AA, 012	53AA	
0AA, 003	14AA	0AA, 013	0AA, 013	54AA	
1AA, 003	15AA	1AA, 013	1AA, 013	55AA	
2AA, 003	16AA	2AA, 013	2AA, 013	56AA	
3AA, 003	17AA	3AA, 013	3AA, 013	57AA	
0AA, 004	20AA	0AA, 014	0AA, 014	60AA	
1AA, 004	21AA	1AA, 014	1AA, 014	61AA	
2AA, 004	22AA	2AA, 014	2AA, 014	62AA	
3AA, 004	23AA	3AA, 014	3AA, 014	63AA	
0AA, 005	24AA	0AA, 015	0AA, 015	64AA	
1AA, 005	25AA	1AA, 015	1AA, 015	65AA	
2AA, 005	26AA	2AA, 015	2AA, 015	66AA	
3AA, 005	27AA	3AA, 015	3AA, 015	67AA	
0AA, 006	30AA	0AA, 016	0AA, 016	70AA	
1AA, 006	31AA	1AA, 016	1AA, 016	71AA	
2AA, 006	32AA	2AA, 016	2AA, 016	72AA	
3AA, 006	33AA	3AA, 016	3AA, 016	73AA	
0AA, 007	34AA	0AA, 017	0AA, 017	74AA	
1AA, 007	35AA	1AA, 017	1AA, 017	75AA	
2AA, 007	36AA	2AA, 017	2AA, 017	76AA	
3AA, 007	37AA	3AA, 017	3AA, 017	77AA	

B3	ADDRESS
020	10000
040	20000
060	30000
100	40000
120	50000
140	60000
160	70000
200	100000

Using the examples above, 000400 translates directly from the first table as 000, 001. Don's example, however, needs two steps: First find the largest segment of 003, 036 which the first table shows (0AA, 016):

$$003, 016 = 7003$$

This leaves 020 unaccounted for in byte 3. The second table shows what to add:

$$\begin{array}{r} 003, 016 = 7003 \\ + \quad 020 = 10000 \\ \hline 003, 036 = 17003 \end{array}$$

The extreme example is 377, 377:

$$\begin{array}{r} 377, 017 = 7777 \\ + \quad 160 = 70000 \\ + \quad 200 = 100000 \\ \hline 377, 377 = 177777 \end{array}$$

Of course we'll all have a program soon to do this for us, won't we? In the meantime, one begins to realize why many users prefer hex:

303, 003, 000 becomes C3, 03, 1E and 17003 becomes 1E03, and the relationship between bytes 2 and 3 and the actual address is once again apparent. No need for tables or programs!

(The use of hex disturbs those who like to see the instruction break-down which octal reflects. Has anybody ever suggested using octal for instructions and hex for addresses (yes, a mixture!) for program writing and assembly printout?)

If you are interested in a Fortran compiler for the Altair 8800 (needs 16K) contact Don Tarbell or Hal Lashlee right away.

HARDWARE

KIT BUILDING TIPS
by Lorin S. Mohler

Since many of us are building kits for the first time and have had little or no experience with electronics, let alone electronic assembly, it might be good to understand a few of the basics on assembling an electronics kit. Let's start with the most important:

TOOLS

It is imperative to have the right tools to do a good job building an electronics kit. Not only will you have an easier job

assembling the kit, but when you are finished, you will have a better chance of it working the first time, and it will probably look a lot neater than without the proper tools. Don't laugh. Neatness counts, because solder bridges are no fun. The following types of tools are recommended. However, whose you buy is up to you.

1. Miniature needle nose pliers with insulated handles. Xcelite 72 or equivalent.
2. Miniature side cutters with insulated handles. Channellock 41 or equivalent.
3. Soldering iron, 42 watt element, iron clad tip. Ungar #1235 element or equivalent. Ungar #7155 tip or equivalent. Ungar #777 handle or equivalent.
4. Solder .031 diameter or smaller. 63/37 or 60/40, resin multicore. Kester or equivalent.
5. Needle tip tweezers.
6. Liquid resin -- should be light amber in color.
7. Tinned braid -- from scrap shielded wire dipped in liquid resin then allowed to dry.
8. Bulb type solder sucker.
9. Soldering iron wiping sponge, any cellulose type.
10. Flux solvent -- isopropyl alcohol.
11. Slot screw driver -- 1/4" wide.
12. Phillips screw driver, #2.

The soldering iron component combination listed gives excellent results and is probably your most important tool.

The iron clad tip, if kept clean and wet, will last a very long time. Do not file this type of tip or its usefulness will be lost. Simply keep it coated with solder when not in use and wipe it on a damp sponge prior to using. Any buildup of dirt on the tip will cause poor solder joints. When the sponge is dirty, throw it away and start with a new one. Washing the sponge won't make it clean enough. When the tip starts to be eaten away at the end, throw it away and replace it with a new one.

You are less apt to get solder

blobs using .031 or smaller diameter solder. It may mean the difference between a good joint and a bridge.

INSTALLING COMPONENTS PRINTED CIRCUITS

1. Bend the component leads to fit before trying to force the part in place.

2. Many component leads will be shaved while being inserted due to a tight fit. Watch for and remove the shavings since they will more than likely find their way to the most damaging short when power is applied.

3. After the component is in place, bend the leads on the back side slightly to hold the part in place. Don't forget you may some day have to remove that part and bending the leads all the way over will only make it difficult to remove. Solder is enough to hold all small parts in place. Larger and heavier components should be cemented to the board.

4. Solder the component in place.

5. Clip off the excess lead length. Do not clip into the solder joint but slightly above it. Hold that lead before you clip or it might wind up in someone's eye or across the power bus in some other piece of hardware.

SOLDERING

1. Always heat the joint to be soldered, not the solder.

2. Feed in the solder slowly using as little solder as possible.

3. Remove the solder then the iron. Leave the iron on the parts only long enough to get the solder to flow freely. On larger areas lay the tip flatter to get more tip to part contact. On large pc areas move the tip around the joint while contacting the board to heat evenly around the lead being soldered.

Solder flows to the heat source so if you get too much solder on a joint, draw the solder blob with the iron tip up the component lead that will later be clipped off. Otherwise refer to the desoldering section.

The joint should be bright and shiny, not dull or cracked, in

appearance. If it is bad reheat it. If the joint won't get shiny add a small drop of liquid resin then reheat. Don't let the components move. Don't feed enough solder to let the joint droop, drip, or bulb. A little solder is all that is needed.

Wash the completed PC board with isopropyl alcohol to remove the resin. Scrub the stubborn resin with an old toothbrush and isopropyl alcohol. Make sure the printed circuit fingers (the part that plugs into the connector) is clean.

DESOLDERING

METHOD A

1. Place a small drop of liquid resin on the joint to be desoldered.

2. Heat the joint with the iron until the solder flows freely.

3. Squeeze the bulb solder sucker.

4. Remove the iron and quickly place the tip of the sucker on the joint. Release the bulb, sucking up the molten solder.

5. Do not apply too much heat or the printed circuit will lift from the board.

METHOD B

1. Place the tinned fluxed braid over the joint.

2. Place the iron on top of the braid heating the joint through the braid. The solder will be sucked into the braid by wicking action. If there is a lot of solder to be removed, gently pull the braid under the iron while in contact with the joint.

Use liquid resin freely, it will help the solder flow. In some cases, it may be useful to add a little more solder then desolder.

REMOVING COMPONENTS

METHOD A

Desolder using either desoldering method. If desoldered properly, the component should come away with little effort. On multilead components such as ICs work each lead loose individually with a sharp pointed tool. Do not pry on body of the part being removed.

METHOD B

If the IC or other component is definitely bad, clip the leads close to the body, remove the carcass, and desolder each lead

individually. This will do a better job of saving the printed circuit.

Do not force a part out by prying on it. Chances are good you will take the printed circuit with it. Do a good desoldering job on each lead.

THE "CPU ON A CHIP" (with digressions) by Roy Fultun

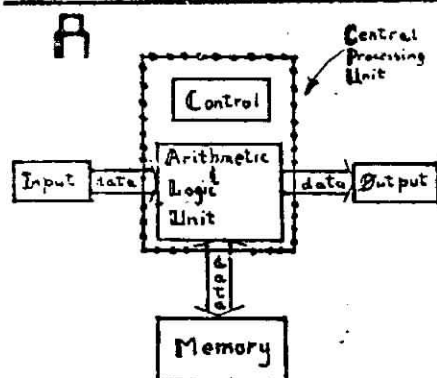
This article attempts to clarify the question of what a microprocessor is and what it does, to introduce functional-block and data-flow diagrams, and to relate these diagrams to currently available microprocessors. In some cases, there seem to be hobbyists who have bought microprocessor-based microcomputers in either kit or assembled form and have little idea of how to apply them even though the computers were bought with a specific purpose in mind.

These poor souls may attempt to arrive at a better understanding of what they've bought by digging through the schematics, only to arrive at a box labelled 'microprocessor'. The microprocessor may be packaged as a single unit or be split into several blocks. But its internal structure remains obscure. While this note will not cover that internal structure, it will shed some light on the kind of thing which "must" be going on inside and should give the reader a foundation from which to relate to the schematics.

Further, the block diagram and data flow approach is general enough to allow a person who has read this note to read the spec sheets of a particular processor with a fair understanding of how the data is handled, manipulated and stored internally by the microprocessor chip.

A: THE FIVE-BLOCK DIAGRAM

The classical five-block diagram of the general-purpose computer (figure A), though slightly oversimplified, does give a reasonably accurate description of the functional arrangement of most common computers from the largest to the smallest. For the sake of simplicity, we will omit the case of "computers" which are in fact networks or arrays of sub-computers. How-



ever, the diagram does tend to apply to the subcomputers.

The five-block diagram is intended to reflect the flow of data through the computer as being a three-step process. From this point of view, data is input to the computer from the outside world, transformed in some manner, and then put back out again.

There is a nice analogy to this process based on the flow of paper across the desk of an office clerk. On his left he has an IN basket. In his hand he has a four-function (pocket) calculator. On the center of his desk he keeps an erasable slate for use as a scratchpad. On his right he has an OUT basket. The rest of the office communicates with him by placing work to be done in the Input basket and retrieving his "answers" from the Output basket. For now it simplifies the discussion to assume that the clerk only processes numerical data even though clerks in the real world are not so limited. For that matter, neither are computers!

The clerk takes slips of paper from the Input basket, records the input data on the slate, decides what to do with it, uses the pocket calculator to perform the calculations (that is, the transformations), uses the slate further to record intermediate answers, and writes his response on a slip of paper which he puts in the Output basket.

To explore the analogy, we note that the Input basket corresponds to the INPUT box of figure A. Concrete examples of an Input unit of a computer include the teletype keyboard, the paper tape reader and the punched card reader.

For those who have bought a microcomputer in either kit or as-

sembled form, and have yet to purchase peripherals such as a teletype or TVTypewriter, the best example of an Input device is the switches (not the lights) on the control console. In the absence of other peripherals the switches are the only input device to the computer. (The control console is a peripheral in all microcomputer systems that bother to use a console.)

The slate would correspond to the memory. We have used a slate rather than paper as an example of a scratchpad because it is easily erasable any number of times. Further, in these diagrams when the term "memory" is used, it is used to refer to Random Access Memory (RAM) which has the property that any piece of data written on it can be accessed as easily as any other. Both instructions and data may be placed in memory and modified during the transformation process.

To complete the analogy, the clerk corresponds to the Control Unit of the block diagram, while the pocket calculator corresponds to the Arithmetic and Logic Unit (ALU). In this context, the word 'Logic' in ALU has less to do with thinking than with arithmetic; the ALU does transformations on data other than simply performing arithmetic operations, and many of these transformations are called logic transformations. Examples would be the AND, COMPLEMENT (equivalent to logical NEGATE), inclusive OR, exclusive EXOR and similar operations found in the list of instructions which the computer can perform. The clerk, together with the pocket calculator, form the analog of the so-called

'mainframe' or Central Processing Unit (CPU) in a concrete computer. The concept of the CPU will be central to the following discussion; in particular, the microprocessor constitutes the CPU of a microcomputer.

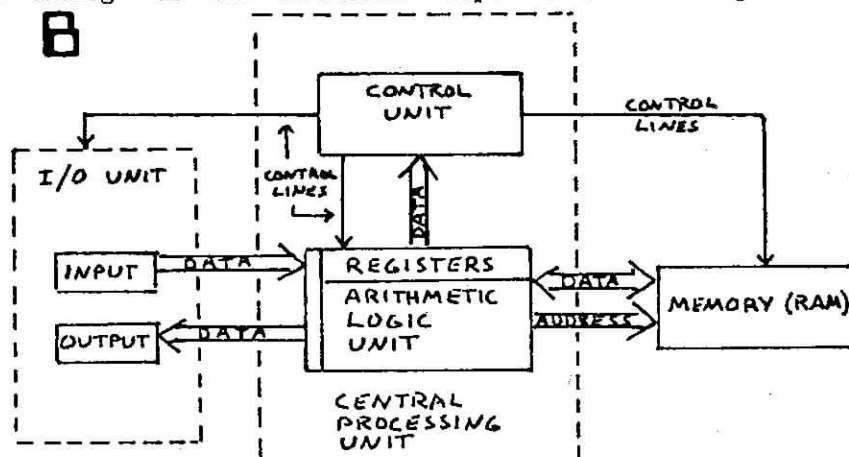
The final element of the analogy lies in giving examples of devices which correspond to the Output basket. For those systems with only a console, the only Output devices are the indicator lights. More useful Output peripherals would be the teletype printer, line printer, or display screen on a TVT.

To summarize: data is input, transformed by the CPU using RAM for temporary storage, and finally output.

B: THE FOUR-BLOCK DIAGRAM

In more modern practice, the five-block diagram might be replaced with a four-block functional unit in which the Input device and the Output device were treated similarly. As a matter of fact, this point of view reflects actual programming practice in which the portions of the program concerned with Input and those concerned with Output are generally written by the same programmer and occupy reasonably adjacent sections of code; the adjacency isn't necessary but can be convenient. These sections of code are referred to as Input/Output or 'I/O' routines.

The four-block diagram has certain advantages over the five-block diagram in that it allows some clarifications about the way the Control Unit fits into the picture, and it allows us to replace the simplistic ALU concept with the more sophisticated



and more realistic concept of the Register, Arithmetic and Logic Unit. The latter is generally referred to as an 'RALU'.

Most minicomputers, and even most bigies, have tended to incorporate a "small" number of memory cells into the same unit as the ALU. Such a memory cell is generally called a register, and the number of registers on an RALU can range from 2 to 256. 4, 8 and 16 are common numbers of registers for various machines. The width (i.e. the number of bits) of a register is usually equal to either the width of the data bus (the word length) or the width of the address buss.

In the four-block diagram of figure B the I/O unit has been depicted as a single box. This occurs in practice in the instance of the teletype (TTY) as well as the TeleVision Terminal (TVT). With the teletype, the keyboard (the Input device) and the printer (the Output device) are packaged in a single cabinet. There is a knob on the front which can be turned into the 'ON LINE' position, the 'OFF' position, and the 'LOCAL' position, in that order. In the 'LOCAL' position, the output of the teletype is connected directly to the teletype printer's input. This yields an expensive typewriter!

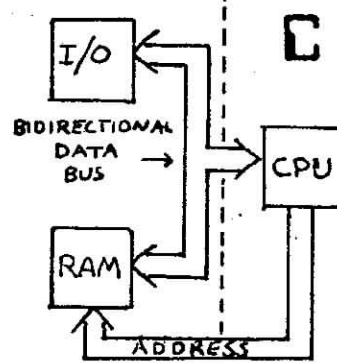
An alternative method is to connect the keyboard and printer as shown in figure B; this is called 'answerback mode'. In answerback mode, as a key is struck on the input keyboard, a binary-encoded character is sent to a register of the CPU or to memory through the Input bus under program control (usually after transversing some interface circuitry). The CPU will then retransmit the received character back out over the Output bus (and then through the interface) to the printer. Answerback mode yields more reliable information transmission. If the typed character is different from the character entered on the keyboard, something is wrong. It also produces a very expensive typewriter! (We won't concern ourselves here with any other methods of using a teletype.)

Notice that we have drawn the Control Unit and the RALU together in a single dotted box.

Taken together, the Control Unit and the RALU (or ALU, as the case may be) form what is generally referred to as the Central Processing Unit or 'CPU'. The memory has been drawn as a separate unit as usual, and through slight simplification can be thought of as being Random Access Memory (RAM) even though memory need not be random access.

Mass memory units such as paper tape readers, tape cassettes and tape drives, disks, drums, and floppy disks are generally treated in much the same manner as I/O devices. I/O programmers are often used to program these devices; mass memory tends to be similar to I/O insofar as it requires special programming attention. This is due to the fact that most I/O devices and almost all mass memory devices are electromechanical, and the consequent timing problems that result cause many headaches (often literally).

I/O units and mass memory differ drastically from true Random Access Memory. With true RAM, in order to store or retrieve information in a memory cell, it is only necessary to supply a (random) address and the control unit will automatically perform the required memory access "immediately". In this case, "immediately" means within one memory cycle, which is generally on the order of magnitude of one microsecond (one millionth of a second). In contrast, the time to read or to write on a particular mass memory unit varies; in the case of rotary mass memory such as drums and fixed-head disks, the average time to access the start of a segment of data is known as the 'latency time' and is generally greater than the main memory cycle by a factor of the order of 1000.



Having defined the term CPU and distinguished it from RAM and I/O, we are now in a position to distinguish between a microprocessor and a microcomputer.

C: MICROPROCESSOR OR MICROCOMPUTER

The distinction made here is arbitrary, not binding on anyone, in conflict with Intel (and possibly other microprocessor manufacturers who want to call their microprocessors 'microcomputers' to emphasize their computational power and small size). It also conflicts with Theodor Nelson's treatment of microprogrammable computers (see Computer Lib, page 44). We are just defining some conventions which the user may follow at his convenience (and risk!); the reader should be aware that a great deal of misunderstanding can arise in discussions with people who use conventions other than the ones outlined here.

The definition used herein for a microprocessor is: a CPU that can be implemented in a small number of integrated circuit packages. Ideally, one package will suffice, and there are a number of brands of microprocessors (micro CPUs) which come in only one package. However, one widely used unit (the National IMP-16) takes 5, and most of the bipolar (TTL or ECL technology, as opposed to MOS) microprocessors take from 6 to 24 to implement the CPU function. If one looks at the NSC IMP-16 (the five chip microprocessor mentioned before), it turns out that it is a 16-bit machine composed of four RALUs, each of which is four bits wide, and an additional control package. Thus it is essentially a 16-bit RALU with a control unit and conforms well to the CPU diagram in the dotted lines of figure A.

The discussion has now progressed to the point where the four-block diagram can be further simplified into a three-block diagram. The three blocks correspond to the three well-defined terms 'I/O', 'CPU' and 'RAM'. The CPU is being treated on a special basis, different from RAM and I/O, as shown in figure C.

For the immediate present, we will use the term 'CPU' to include the card on which the microprocessor package or packages

are mounted. This card will contain support packages in addition to the microprocessor package; however, the overall function of the card corresponds to what we would ideally put in a single package if it were practical to do so.

The discussion is now leading towards the definition of the term 'microcomputer'. That definition must be temporarily postponed while the structure of a microcomputer is explored.

This exploration will make use of the block diagram in figure D. For the remainder of this section, the discussion will be card-oriented and the functional blocks inside the microcomputer will correspond to cards which take on the functions described in blocks occurring in previous diagrams. The case of the RAM card is easy, since several RAM cards together perform the function represented by the RAM block diagram.

The busses of the microcomputer are the Control Bus, the Address Bus and the Data Bus. These connect into the ports of the various cards through edge connectors. The data and address busses work exactly as described before internally to the microcomputer. The control wires, which technically should be considered as a bus, have been ignored in most of the previous diagrams for the sake of simplicity; nevertheless, they have been implicit in all diagrams in which they have not been explicitly shown.

The I/O cards can be considered mediator cards between the outside world, i.e. the peripherals themselves, and the microcomputer. They provide the interface function between the peripherals and the microcomputer in the sense that data and control external to the microcomputer take on forms different from the internal bus structure. Much the same is true for external addresses.

It should be noted that the peripherals themselves are being treated as separate entities from the microcomputer or any part thereof.

Having previously defined 'microprocessor', and having discussed the functions and implications of the dotted block in

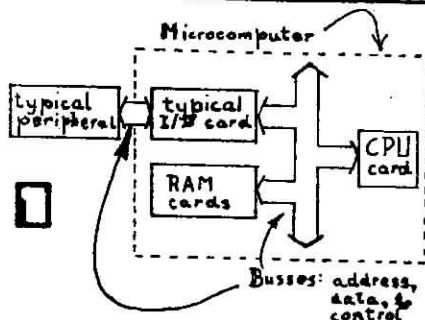


figure D, we are now free to define a microcomputer as a mini-computer built around a micro-processor.

D: BUFFERING

By now it has become appropriate to emphasize that in the general case, signals such as the signals on a data bus, those on an address bus and the control signals to and from the CPU, appear at different current strengths and different voltages at various points of a given system. Interface circuitry will mediate the same signal as it appears at various points and in various forms throughout the system.

In particular, the CPU will have been made using Metal Oxide Semiconductor technology (MOS); hence it will belong to one of the MOS families of logic devices (such as CMOS, NMOS, PMOS, SOS etc). One of the characteristics of MOS devices is that they usually cannot supply powerful signals. This is often expressed by saying that they have insufficient drive capability or, equivalently, that they have low fanout.

(The reader uninterested in electrical considerations may scan on to the next point of interest.) Drive capability and fanout are relative concepts, but there is a common reference standard centered on the TTL (Transistor-Transistor Logic) family. TTL logic devices operate on a 5 volt power supply. All TTL packages have a ground pin which is customarily assigned the level of 0 volts, while they have a power pin (Vcc) which is connected to a power supply bus (not the same thing as a data bus) carrying a potential of 5 volts above ground. Logic signals must fall into certain voltage ranges to qualify as valid signals. For TTL, any input voltage between +0.4 volts and ground (0 volts) qualifies as a logical '0' or

'LOW'. Similarly, any input voltage between 2.4 volts and 5 volts qualifies as a valid '1' or 'HIGH'. Typical TTL output voltages are 3.3 volts for a HIGH value and 0.22 volts for a LOW value. Note that these voltages are valid in the sense that they fall well within the range of valid TTL input levels as just specified.

Suppose we have an MOS output line which has no load attached. (This is sometimes expressed by saying that the load has "infinite" impedance i.e. "infinite" resistance and zero conductance.) In particular, such a load, i.e. no load at all, will sink zero current and the output voltage will not be pulled down toward ground.

The typical TTL output is required to be able to source 400uA and to sink 1600uA in order to drive ten TTL input loads. In order to drive a single TTL input, that gate would have to source 40uA and to sink 1600uA. An input which requires such driving capacity is said to be equivalent to one standard TTL load.

This turns out to be a bit much for some of the early MOS outputs (circa 1970). Back in the bad old days of the early 1970's, it was usually necessary to connect a high-impedance sense amplifier (for the purpose of sensing voltage levels) on the output pins of MOS packages in order to boost their signals to a level strong enough to drive at least one standard TTL load. Generally, it is just as easy to make the output gate of the sense amplifiers strong enough to drive a little over ten standard TTL loads, and so to duplicate the fanout (a little over ten) of the standard TTL driving gate.

It is now relatively easy to build MOS devices with "TTL compatible outputs". Whenever you hear that phrase, be a little wary because it may indicate a power driving capability anywhere in the range just sufficient to drive a single Low Power TTL input (which is considerably less than a standard TTL input) clear up to enough power to drive slightly over two standard TTL inputs. The phrase 'fully TTL compatible'

generally indicates at least enough power to drive a single standard TTL input. With some of the manufacturers producing quality devices, it can indicate a fanout of at least two standard TTL inputs. This latter capability is handy insofar as it allows the output to drive a large number of MOS inputs while still retaining the capability of driving a standard TTL load which is used to boost the signal to standard TTL fanout using a standard TTL device.

The process of using an intermediate device to increase fanout is known as (power) buffering (not to be confused with data buffering) and is an extremely useful trick of the trade in logic design. It is so useful that you may rest assured that it is used on your microprocessor card. The only exception to this rule is the case of the multiple-package bipolar microprocessors in which the chips internal to the microprocessor packages have outputs sufficiently powerful to drive a large number of (TTL) inputs.

It should be pointed out that there is an implicit separation between the data lines at the input of the buffers and the data lines at the buffer outputs as outlined heretofore in this discussion. While the techniques can be used profitably in design work, it is not an inherent feature of the buffering process.

E: BUSSING

One important aspect of the block diagrams that has been ignored in the prior exposition is the question of data flow on busses. It has been assumed so far that addresses flow out of the CPU only. This is in fact the case in most systems; such a system would have a unidirectional address bus. While this type of bus is consistent with buffering (remember that the address output lines of that weak little microprocessor just can't drive lots of TTL loads), it is not the only way to go. Heh Heh.

Buffering is not inconsistent with bidirectional busses. The data busses on the RAMs in all figures of this article have been depicted as being bidirectional. While RAMs need not

have bidirectional busses, it is usually the case that they are implemented in this way for most system-level RAMs which hang off the data bus. As a matter of fact, it is usually easier to implement systems in which the data bus is bidirectional.

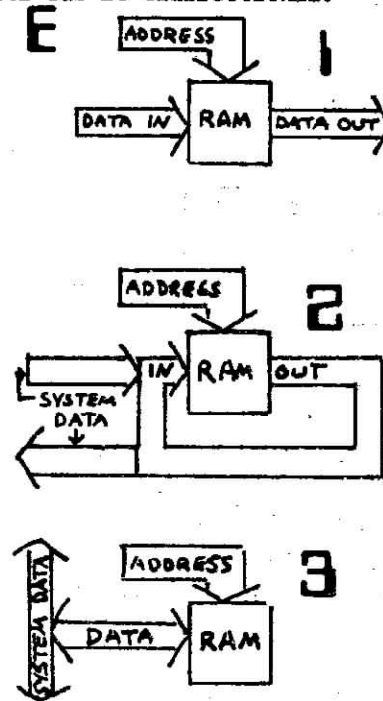


Diagram 1 of figure E shows the structure of some early TTL RAM devices such as the 7489 and the 74200. The inputs and outputs are separate. Diagram 2 is a modification of diagram 1 and reflects corresponding interconnections which can be performed with the above devices without breaking any design rules. The system data inputs and outputs are shown (and are shown separately) for the purpose of introducing the rest of the system into the discussion in a manner consistent with diagram 1. Diagram 3 shows the overall effect of this process.

Bidirectional bussing is not possible with the standard TTL logic gate which has an output structure with ability to powerfully pull its outputs either HIGH or LOW; the output gate cannot be in any other state. There are two methods for implementing bidirectional bussing by means of modifying the structure of the standard TTL output gate. The following discussion will use the term 'bus' to indicate a single signal.

The oldest, known as the open collector gate (O.C.), is available and common in all of the

TTL logic series offered by most manufacturers. O.C. logic will not be discussed here, since it is rapidly becoming obsolete.

Tristate is much like standard TTL except that many outputs may be connected together. There is a third state (other than HIGH or LOW) which all except one of the gates must be in. This third state is one in which the gate goes into high impedance (high-Z) and is unable to pull the bus either HIGH or LOW; such an output is effectively removed from the bus. Not more than one gate may be enabled at a given time. That gate then functions precisely like a standard TTL gate.

The reader is referred to Don Lancaster's TTL Cookbook for a more detailed discussion of these matters. Many of the subtleties of the schematics describing the circuits of the microcomputer of your choice will revolve around the properties of the open collector and tristate circuits precisely because it is through these mechanisms that bidirectional bussing is implemented.

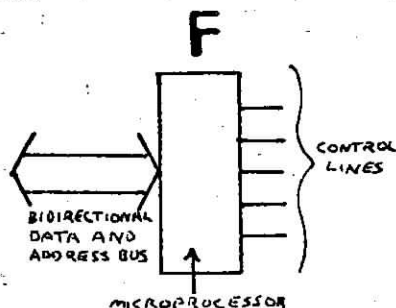
F: A VERY CONDENSED SURVEY OF CURRENTLY AVAILABLE MICROPROCESSORS

The first candidate for the title of "computer on a chip" was the Intel 4004 introduced privately prior to 1971 to a manufacturer for use as a controller in its table-top business calculator. Whether through accident or more likely intent, this 4-bit machine technically qualifies as a bona-fide microprocessor and hence as the CPU for a bona-fide general purpose microcomputer. It is not very practical for applications other than those for dedicated special purposes such as Point-of-Sale terminals, automatic gas pumps or early generation non-programmable calculators. An improved version, the 4040, sports an expanded instruction set and includes a few of the standard logical instructions which are such a crutch to programmers and are so conspicuously missing on the 4004.

Due to limited space, the following discussion utilizes a system-level classification according to external port arrangement, equivalent to des-

cribing the bus arrangement from the point of view of the package pinout. The 4040 and the Fairchild F8 have been omitted, the former on the grounds of insufficient ease of use and insufficient generality. The F8, which shows great promise for the future, has not currently received adequate development toward use as a hobbyist machine. Further, the F8 stands unique in terms of our classification system.

Intel was the first to introduce a practical microprocessor which emerged in late 1971 in the form of their 8008. The 8008 is an 8-bit MOS microprocessor built in an 18 pin package. It has the bus structure of figure F. In this arrangement, a single 8-bit port is time-multiplexed in the sense that one portion of the instruction cycle time is reserved for putting out the six high-order address bits, another is reserved for outputting the 3 low-order address bits, and other portions of the cycle are reserved for inputting and outputting data to and from the RAM.



With the Intersil IM6100, a CMOS microprocessor, the situation is simpler. The 6100 recognizes the instruction set of the PDP-8 and does a good job of running the software and driving the peripherals of that machine. Of all the microprocessors, this machine alone has a 12-bit data word and a 12-bit address bus. Due to the equality of the widths of the data word and the address bus, figure F can be used to describe this machine. The single 12-bit bus can be used to output addresses, output data, or input data at appropriate portions of the instruction cycle.

The National PACE, as well as the National IMP-16, which are almost identical from the point of view of their instruction sets, also time-multiplex their single bus in the same general

manner as the IM6100. Still conforming to class F, they have a 16-bit port.

The General Instrument IP1600 is equivalent to the National machine insofar as it falls into class F and has a 16-bit port. Of course, it recognizes an entirely different instruction set and handles I/O control in a different manner.

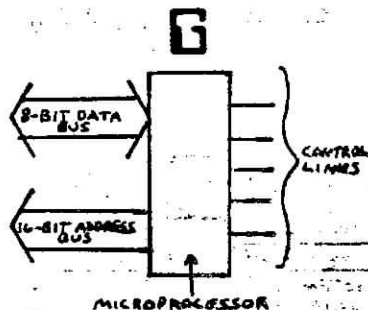


Figure G is used as the basis of the bussing scheme for the Intel 8080, the Motorola MC6800, the Signetics 2650, and the MOS Technology MCS6501 microprocessors. While these microprocessors have the same general bussing arrangements, there are considerable differences between them in the following respects:

1. Internal architecture and the number of on-chip registers vary.
2. The instruction set, including its relationship to the architecture also varies. Not only do the number of internal registers differ, but their widths, internal bussing, and their interactions (such as coupling two 8-bit registers together to make a single double-precision register -- or the failure to do so) differ with strong repercussions on the relative information-processing powers of the different machines.

3. The control lines and the way they are used in conjunction with peripheral chips and packages varies.

4. The Signetics 2650 has completely static clock operation in the sense that the clock transitions are level-sensitive and the registers will retain their information even when the clock halts at a (DC) logic '0' or '1'. The only other microprocessor possessing this capability is the Intersil IM6100 (think PDP-8).

5. The IM6100 has the added advantage of being able to use all CMOS system components. CMOS is a logic family similar to TTL. Its fanout is just barely one or two standard TTL loads when driven with a 5 volt supply, but its inputs have extremely high input impedance and do not tend to load the outputs much at all.

6. The MOS Technology 6501 does not have a tristate address bus, and in that sense is unique in class G. Even in the case of the processors with tristate address busses, data flow is unidirectional out (data does not flow into the processor over the address bus). For those who want to use the 6501 with a tristate address bus, it is a simple matter to use four 74S257 quad 2-to-1 tristate multiplexers. We leave it to the reader to conjure up uses for the extra input bus to the multiplexer.

!! !! !! !! !!
(note: Roy's cooperation in getting this article out in record time as a last-minute substitution is greatly appreciated. I personally apologize for putting him through this rush, but I'm sure you'll agree that the results were worth it! Jon)

MANUFACTURER & PART NUMBER	DATA BUS WIDTH	ADD. BUS WIDTH	BUS CLASS (fig)	MFG. PROC.	POWER SUPPLY LEVELS	NO. PINS
Intel 4004	4	12	F	MOS	-10,0,+5	16
Intel 8008	8	14	F	MOS	-9,0,+5	18
Intersil IM6100	12	12	F	CMOS	0,+5 to +10	40
National IMP-16	16	16	F	PMOS	-12,0,+5	120
National PACE	16	16	F	PMOS	-12,0,+5	40
G.I. CP1600	16	16	F	NMOS	-12,0,+5,+12	40
Intel 8080	8	16	G	NMOS	-5,0,+5,+12	40
Motorola MC6800	8	16	G	NMOS	0,+5	40
MOS Tech. MCS6501	8	16	G	NMOS*	0,+5	40
Signetics 2650	8	16	G	NMOS*	0,+5	40

*ion implanted

*****HELP!*****

HELP!! I need someone to help me set up an IBM 7094 with 300K of memory. I will give computer time for your help. No experience is necessary. If interested call Alex Szerlip at (213) 325-9333.

HELP!! My Altair 4K RAM board won't remember a single byte. If you have any experience or suggestions for these boards, please contact: Rudy Hirschman, 1001 Kagawa St, Pacific Palisades, Ca. 90272, (213) 454-1277

*****INSTRUCTION*****

Are you interested in a construction class (days and/or nights) in using ICs and discrete components? Reserve registration PO Box 1260 South Gate 90280.

*****COLLABORATION*****

Would like to talk to a programmer on real estate programs. Call Owen Sherrill at 349-1140.

I am looking for someone in the Venice-Santa Monica area who is planning to build an Altair kit. If you let me watch, "help", and ask lots of questions while you assemble and debug the hardware, I'll spend an equitable amount of time writing programs for

AOS

your machine once it is up. I understand hardware only at the functional level, but have quite a bit of experience programming in assembly and higher level languages. Call Larry Press (213) 399-2083.

*****SELL OR SWAP*****

"What To Do After You Hit Return or PCC's First Book of Computer Games" -- list price \$6.95. Hal Lashlee picked up a limited number from PCC. If you call or write Hal before Sept. 15 and you purchase before the start of the meeting on Sept. 20, the price is \$5.00 including tax.

CASSETTE INTERFACE FOR 8800: 187 bytes/sec, 800 bits/in, up to 500K bytes on C-60, phase encoded -- self clocking, uses only one channel, interfaces to any good audio cassette unit with no mods, uses low noise audio cassettes; 4 extra status lines available for input, 4 buffered control lines available for output, device code selectable with dipswitch. Basic loads in less than 40 sec. instead of 12 min. \$100 complete kit (cassette not included). Don Tarbell 832-0182

RPC-4000 computer with punch, reader and typewriter. Best offer over \$400. Two old scopes \$25 each. Three nice cores \$15 each. 8X1K core memory with PS \$35. Varian chassis with power supply \$50. Teletype reperforator with KB \$15. Wyle processor w/KB and display \$35. Several readers, printers, tape rewinds \$15 more or less. Call Bill Pfeiffer at (213) 796-8270.

TAPE DRIVE: BW 301-A, 9-track, 800 BPI, 110 VAC, 1 phase, 60HZ. Missing parts: 2 cards, 5v regulator, button panel. Complete documentation. 25 ips. \$250. Don Tarbell -- 832-0182.

POWER SUPPLIES: I have a quantity of 5v-10 amp highly regulated power supplies taken from keyboard terminals. They also supply 200v, 12v and 48v each at 1 amp. I will provide schematics and plans for obtaining -5v, -9v and -12v. \$25 plus postage on 15 lb and 6% in Calif. Grant Runyan, 1146 Nirvana Rd, Santa Barbara, Ca 93101. 805-962-7734

*****WANTED*****

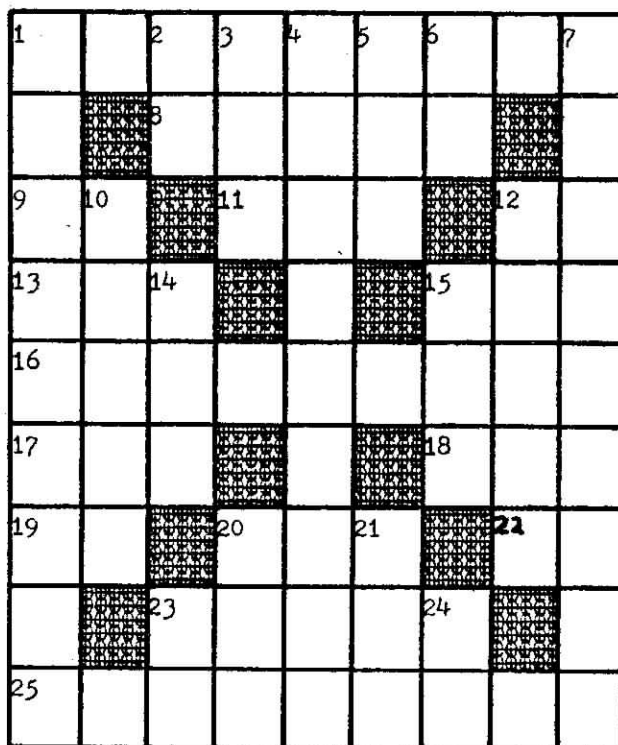
If you (1) have a PROM programmer, (2) can get my TTY to type zeros, or (3) can do good PC work, please contact Don Tarbell at 832-0182.

THE INTERFACE INTERLACE

... a puzzle to weave your way through...

ACROSS:

1. Someone with Basic who won't share it!
8. High in the air.
9. Millifarad (abbr).
11. A plot of land.
12. Chopping tool.
13. The centimeter-gram-second unit of work.
15. Australian flightless bird.
16. One who enters data by the front panel.
17. Said at weddings (two words).
18. Black, tropical American cuckoo.
19. Arsenic (abbr).
20. Arrival (abbr).
22. Georgia (abbr).
23. Village of South African natives.
25. Hardware



DOWN:

1. Data included in a machine language program.
2. SCCS headquarters.
3. Sick.
4. Machine routine to get a computer up.
5. A newt.
6. Right (abbr)
7. Richly abundant.
10. Astair, McMurray and Flintstone.
12. My Souvenirs.
14. Government Printing Office (abbr).
15. 7th letter of the Greek alphabet.
20. Nickname for Greek shipping magnate.
21. Random access memory.
23. University in Lawrence, Kansas.
24. French article.

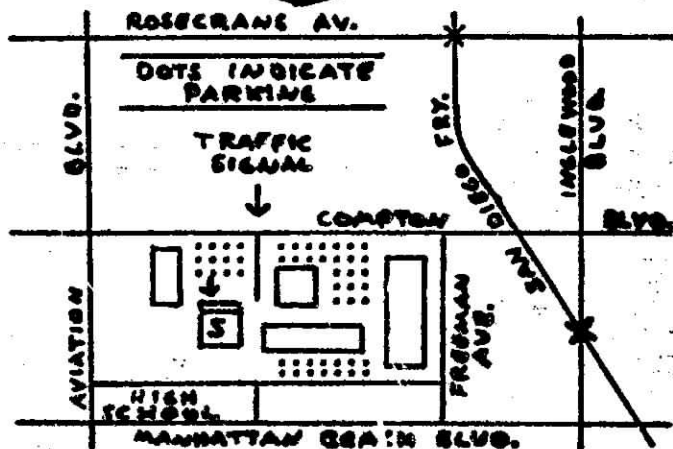
NOTE: THE MEETING DATE
ON THIS PAGE IS CORRECT.
REFERENCES INSIDE TO THE
20th ARE IN ERROR!!!!!!

MEETING
SUNDAY, SEPT. 21, 1975

BUILDING "S" CAFETERIA
TRW SYSTEMS
REDONDO BEACH

DOORS OPEN AT 10:00 AM
COFFEE HOUR 12:00-1:00
MEETING 1:00-4:30

BRING YOUR EQUIPMENT TO
SHOW OFF, SELL OR SWAP.



MEMBERSHIP FORM

SOUTHERN CALIFORNIA COMPUTER SOCIETY
P.O. BOX 987
SOUTH PASADENA, CA. 91030
(213) 682-3108

NAME

ADDRESS

CITY

STATE

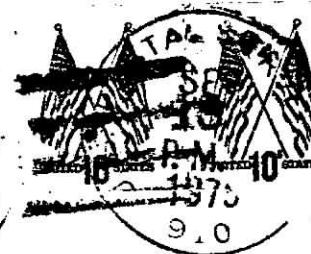
HOME PHONE

BUSINESS PHONE

DUES: \$10.00/year

PLEASE PRINT PLAINLY

INTERFACE
11557 SUNSHINE TERRACE
STUDIO CITY, CA. 91604



FIRST CLASS MAIL

STD RESEARCH CORP
POST OFFICE BOX "C"
ARCADIA, CA. 91006

ATTN: CHARLES MS KINNON

RECEIVED
SEP 16 1975
STD RESEARCH CORP